manuale

4

Per i possessori di computer

ZX SPECTRUM e TI 99/4A

PSEUDOCODICE:

- Strutture dati
- Coda lista
- Simulazione

I SISTEMI DI NUMERAZIONE

LE PERIFERICHE DI INGRESSO E USCITA

I CICLI DEL BASIC

EV EDITORIALE VIDEO

COMPUTING VIDEOTECA fino a questo momento ha offerto ai suoi lettori le lezioni di Pseudocodice, un Corso di Basic per lo ZX SPECTRUM, articoli-guida alla programmazione, notizie importanti sui segreti del vostro computer, trucchi per personalizzare i vostri programmi e tante altre cose che, riteniamo confortati dalle vostre migliaia di lettere e telefonate, abbiano reso piuttosto indispensabili questi manuali periodici destinati ad affiancare il vostro prezioso ZX SPECTRUM.

L'entusiasmo dei lettori ci spinge a offrire sempre di più e perciò vi annunciamo con piacere che, a partire dai prossimi numeri, COMPUTING VIDEOTECA si arricchirà anche di lezioni chiare e indispensabili sul Linguaggio Macchina dello ZX SPECTRUM.

Purtroppo le lezioni di linguaggio macchina non servono per il TI II/4A, comunque i possessori di questa duttile e intramontabile macchina potranno consolarsi con gli articoli sullo pseudocodice, sui sistemi di numerazione e sulle periferiche che sono tutti utilissimi.

Se i manuali riscuotono un grande successo, entusiasmo non certo inferiore desta la raccolta di sfotware in cassetta con cinque giochi e programmi destinati ai possessori di ZX Spectrum e altri cinque per coloro che possiedono il TI 99/4A.

Per lo ZX Spectrum i programmi sono: Mixxil (un bellissimo gioco di strategia), Dizionario (utile per tradurre moltissimi termini tra italiano e inglese) Sequencer (gioco di abilità e memoria) Lines (un programma divertente che metterà a dura prova la vostra abilità) e infine un tradizionale e utilissimo programma di Bioritmi.

Per il TI 99/4A i programmi sono: Space War (gioco spaziale) Slot Machine (un programma molto divertente che vi permetterà di scommettere con voi stessi) Agente segreto (gioco di abilità) Dieta e Calorie (due programmi utilissimi per nutrirsi meglio e divertirsi).

Per i nuovi lettori ripetiamo le indicazioni per ottenere gli arretrati: è sufficiente inviare la richiesta (specificando i numeri desiderati) su vaglia intestato a Editoriale VIDEO - Via Castelvetro 9 - 20154 Milano. Oppure si può spedire la richiesta, al medesimo indirizzo, allegando un assegno. L'importo di ciascun numero arretrato, comprensivo di tutte le spese di invio, è di 15.000 lire.

Ringraziamo ancora i lettori e in particolar modo tutti coloro che ci comunicano suggerimenti e idee e collaborano quindi a rendere sempre più piacevole il nostro lavoro.

COMPUTING VIDEOTECA N. 4 - Editoriale VIDEO - Direttore: Antonio Lucarella - Coordinamento tecnico: Roberto Treppiedi - Hanno collaborato: Marco Affer, Alberto Barbati, Massimo Cellini, Umberto Colapicchioni, Umberto Fassi, Alfredo Malgrati, Carlo Mantegazza, Stefano Milanesi, Maurizio Monteverdi, Sebastiano Pastore, Glauco Piatteletti, Daniele Riefoli, Alessandro Vallone, Giovanna Zampella - Stampa: La New Graf Milano - Fotocomposizione: ERREGI Milano - Reg. del Trib. di Milano n. 519 del 10/11/84 - Stampato in Italia

• STRUTTURE DATI COMPLESSE
• CODA LISTA
• SIMULAZIONE

Si definiscono SEMPLICI, le strutture dati che sono realizzate con una esplicita dichiarazione all'interno di un linguaggio di programmazione. In BASIC sono semplici la variabile singola (per esempio A, oppure A$), la variabile con un indice (per esempio A(100)), e le variabili con due o più indici (per esempio A(10,20)).
In altri linguaggi, per esempio il COBOL ed il PASCAL, esiste la variabile strutturata a RECORD, che è definita come un insieme di variabili sia alfanumeriche che numeriche.
Si definiscono "strutture dati COMPLESSE", quelle organizzazioni di variabili tali che non possono essere dichiarate esplicitamente nel linguaggio di programmazione in oggetto.
Un esempio che esamineremo in questo articolo è la struttura di LISTA e la struttura di CODA.
La coda è una struttura dati che non è considerata "primitiva" in nessun linguaggio di programmazione, dichiarabile per esempio con la frase: A(100) CODA.

LA LISTA

Il fatto è che per esprimere certi problemi, o per descrivere certi algoritmi, è molto comodo immaginare che i dati siano organizzati in modo complesso.
Facciamo un esempio. Supponiamo di voler scrivere un algoritmo che memorizza la lista della spesa da fare. È semplice per noi dire: "aggiungere un articolo alla lista", oppure "eliminare un articolo dalla lista", oppure "correggere la quantità da comprare di un certo articolo della lista". In verità noi stiamo facendo riferimento alla parola "lista" perché essa è un modo ben conosciuto di organizzare i dati, le cui caratteristiche (a prima vista ovvie) sono immediate.
Analizziamo cosa intendiamo per lista:

1. I dati sono raccolti in modo che siano uno di seguito all'altro.
2. La lista ha termine dopo l'ultima riga memorizzati, oppure quando termina lo spazio fisico riservato per la lista.
3. Si suppone che inserire un elemento in una lista significa aggiungere un nuovo articolo in fondo alla lista.
4. Si capisce subito che per eliminare un elemento dalla lista è sufficiente fare in modo che esso sia "saltato" o "ignorato" quando si scandisce la lista, per esempio mettendo una croce sopra.

Ecco perché è così importante per noi usare il concetto astratto di lista, invece che spiegare esplicitamente i vincoli a cui

deve sottostare un elenco di variabili per essere chiamato "lista"!
Per il conoscitore del BASIC è ovvio (forse fin troppo), che la struttura dati "variabile con indice" realizza BENE una struttura di lista. Ed infatti la variabile con indice è la "materia prima" con cui organizzare una "lista". Ma la struttura da sola non basta: è importante anche il modo di GESTIRLA affinché si realizzi una vera struttura di lista.
In realtà per organizzare una lista servono due strutture dati semplici: una variabile ad indice A$ ed una variabile numerica P, che serve per memorizzare l'indice dell'ultimo elemento della lista memorizzato.
Con lo pseudocodice possiamo esprimere gli algoritmi di gestione di una lista, che sono:
— inserimento dell'elemento E$ nella lista [INLISTA]
— eliminazione dell'elemento E$ dalia lista [DUTLISTA]
— scansione (e stampa) della lista [SCANLISTA].
Supponiamo di dimensionare la lista a 100 elementi:

```
INIZIO. INLISTA.
  SE P+1>100
  ALLORA
    Scrivi "Non v'è più spazio
    nella lista per l'elemento"
    E$
  ALTRIMENTI
    P=P+1
    A$(P)=E$
  FINE_SE
FINE.
```

```
INIZIO. OUTLISTA
  J=1
  RIPETI
  FINCHÉ (J≤P) AND (A$(J)≠
  E$)
    J=J+1
  FINE_RIPETI
  SE J≥P+1 ALLORA
  Stampa "Elemento non
  esiste nella lista"
  ALTRIMENTI
    RIPETI PER J DA J A P
      A$(J)=A$(J+1)
    FINE_RIPETI
    P=P—1
  FINE_SE
FINE.
```

La OUTLISTA cerca l'elemento richiesto per l'eliminazione (che supponiamo memorizzato in E$) nel primo ciclo di ripetizione. Poi quando il ciclo si interrompe, se J è maggiore di P+1 significa che abbiamo esaminato tutta la lista senza trovare l'elemento da eliminare, altrimenti si procede a far salire di una posizione ogni elemento della lista dal j-esimo in poi (istruzione A$(J)=A$(J+1)).
L'ultima procedura che esaminiamo è la procedura di SCANSIONE della lista: un algoritmo che emette tutti gli elementi "vivi" della lista:

```
INIZIO. SCANLISTA
  RIPETI
  PER J DA 1 A P
    Stampa A$(J)
  FINE_RIPETI
FINE.
```

Osservate che il puntatore P è stato aggiornato ad ogni inserzione e ad ogni eliminazione e perciò conserva sempre l'indice dell'ultimo elemento della lista.
Il modo migliore per comprendere il funzionamento della lista è quello di provare a giocare col programma BASIC di figura 1.
Con questo programma interattivo provate ad inserire nomi di

persone nella lista, ad eliminarle e poi a stampare la lista aggiornata. In breve tempo comprenderete l'importanza di questa struttura dati e sarete pronti per usare gli algoritmi qui indicati all'interno dei vostri programmi.

La LISTA è una struttura dati complessa, realizzabile mediante una variabile con indice ed un opportuno corredo di algoritmi di gestione. Questa affermazione è vera in generale: "OGNI STRUTTURA DATI COMPLESSA È REALIZZABILE A PARTIRE DALLE STRUTTURE DATI SEMPLICI, SU CUI SONO BASATI I PROGRAMMI DI GESTIONE DELLA STRUTTURA STESSA".

In generale i programmi di gestione sono sempre:

- "inserimento di un nuovo elemento"
- "eliminazione di un elemento"
- "ricerca di un certo elemento"
- "stampa di tutta la struttura", (detto anche "visita ordinata degli elementi della struttura")

Quante strutture dati complesse sono state inventate?

Per rispondere alla domanda è necessario riflettere un momento: è chiaro che chiunque può inventare, per i propri scopi particolari, la struttura dati che gli risulta più comoda. Solo alcune, però, di tutte le strutture possibili, sono di carattere così generale da meritare uno studio ed un nome ben conosciuto e divulgato. Tra di esse le più fondamentali sono:

— Liste lineari semplici (quella appena studiata)
— Coda
— Lista ad anelli/doppia/circolare
— Pila
— Alberi
— Grafi

Le liste di ogni genere, le code e le pile rientrano tutte nella categoria delle strutture dati lineari. Gli alberi ed i grafi sono invece strutture non lineari, e verranno affrontati in un altro articolo.

LA CODA

Analizziamo invece una struttura dati complessa altrettanto ovvia quanto la lista, e cioè la "coda".

Immaginiamo di stabilire una organizzazione di dati tale per cui l'ultimo elemento della struttura sia anche l'ultimo elemento estraibile dalla struttura. Abbiamo realizzato in modo astratto una struttura dati che riproduce la sequenza di arrivi e partenze di persone (od oggetti) da una coda, dove per ARRIVO si intende inserimento di una persona in fondo alla coda, e per PARTENZA si intende l'eliminazione della prima persona in testa alla coda.

Perciò una coda è descritta altrettanto bene dalle due frasi seguenti: Una coda è:

a) Una lista in cui le eliminazioni riguardano solo l'elemento in testa alla lista, e l'inserimento avviene in coda.
b) Una lista di elementi con accesso FIFO (dall'inglese FIRST IN, FIRST OUT, cioè: il primo ad entrare è anche il primo ad uscire).

Come si realizza una coda? Naturalmente la materia prima è ancora una volta una variabile con indice, per esempio Q$(1ØØ). Oltre a questo sono necessarie **due** variabili numeriche: una

per conservare l'indice all'elemento di testa della coda, e la seconda per conservare l'indice del fondo della coda. Chiamiamo il primo Ted e il secondo F.
Abbiamo bisogno ora di 2 algoritmi di gestione: INCODA che inserisce un elemento in coda, ed OUTCODA che elimina l'elemento di testa della coda.
Supponiamo di avere una variabile con indice Q$(100) di cento elementi come supporto della coda. Inizialmente T = F = O indica che non c'è nessun elemento in coda. Man mano che arrivano elementi, F si sposta sempre più verso il numero 100. Intanto alcuni elementi si eliminano dalla testa della coda ed anche T si sposta verso 100.
Se gestissimo la coda in questo modo banale, vedremmo che quando F è arrivato a 100, e T per esempio a 30, non potremmo introdurre un nuovo elemento in coda (perché F non può arrivare a 101) sebbene noi abbiamo 29 posizioni libere in testa al vettore. Per questo il modo migliore è gestire la coda con una lista circolare: dopo il 100 elemento, se c'è posto, l'inserimento ricomincia da 1.
Detto questo, passiamo agli algoritmi:

```
INIZIO. INCODA
 FL = F
 SE FL = 10 ALLORA FL = 1
 ALTRIMENTI FL = FL + 1.
 SE FL = T
 ALLORA
  Stampa "non ci stanno
  altri elementi in coda"
 ALTRIMENTI
  F = FL
  Q$(F) = E$
  SE T = O ALLORA T = 1
 FINE_SE
FINE

INIZIO. OUTCODA
 SE T = 0
 ALLORA
  Stampa "Nessun elemento
  in coda
 ALTRIMENTI
  E$ = Q$(T)
 FINE_SE
 SE T = F
 ALLORA
  F = O
  T = O
 ALTRIMENTI
  SE T = 100
  ALLORA
   T = 1
  ALTRIMENTI
   T = T + 1
  FINE_SE
 FINE_SE
FINE
```

L'algoritmo di visita della coda è comprensibile se si pensa che i casi sono due: o la lista comincia da T e finisce ad F (caso in cui T ≤ F), oppure la lista va da T alla fine del vettore, e poi gli ultimi elementi vanno da 1 a F (caso in cui F < T). (Questo ultimo caso è quello descritto prima, in cui si fa uso della ciclicità della lista).

```
INIZIO. SCANCODA.
 Stampa "INIZIO CODA"
 SE T ≤ F
 ALLORA
  RIPETI PER J DA T A F
   Stampa "CLIENTE" Q$(J)
  FINE_RIPETI
 ALTRIMENTI
  RIPETI PER J DA T A 100
   Stampa "CLIENTE" Q$(J)
  FINE_RIPETI
  RIPETI PER J DA 1 A F
   Stampa "CLIENTE" Q$(J)
  FINE RIPETI
```

FINE_SE
FINE.

Come nel caso della lista, anche per la coda diamo un programma BASIC col quale provare gli algoritmi descritti. Buon divertimento e...arrivederci.

```
10 INPUT"INTRODUCI E,I,S (OPPURE F PER FINE)   ";C$
20 IFC$><"S"THEN40
30 GOSUB210:GOTO100
40 IFC$><"E"THEN60
50 INPUT "INTRODUCI ELEMENTO DA ELIMINARE";E$:GOSUB140:GOTO100
60 IFC$><"I"THEN90
70 INPUT"INTRODUCI ELEMENTO DA INSERIRE          (FFFF PER FINIRE)";E$
80 IF E$="FFFF"THEN90
85 GOSUB110:GOTO70
90 IFC$="F"THEN105
100 GOTO10
105 STOP
110 REM INLISTA ***********************
120 IFP>9THENPRINT"NON SPAZIO":RETURN
125 P=P+1:A$(P)=E$
130 RETURN
140 REM OUTLISTA **********************
150 J=1
160 IF(J>P)OR(A$(J)=E$)THEN185
170 J=J+1
180 GOTO160
185 REM FINE RIPETI
190 IF J>P+1THENPRINT"ELEMENTO" E$ "NON ESISTE NELLA LISTA":RETURN
195 FORJ=J TO P:A$(J)=A$(J+1):NEXTJ:P=P-1
200 RETURN
210 REM SCANLISTA *********************
211 PRINT"**** INIZIO LISTA  ****"
220 FORJ=1TOP:PRINTA$(J):NEXTJ
251 PRINT"****  FINE LISTA ****"
260 RETURN
READY.
```

```
5 DIM Q$(10):T=0:F=0
10 INPUT"INTRODUCI A,P,S (OPPURE F PER FINIRE) ";C$
20 IF C$><"S"THEN40
30 GOSUB210:GOTO100
40 IF C$><"P"THEN60
50 GOSUB140:INPUT"SE NE VA IL CLIENTE";E$:GOTO100
60 IF C$><"A"THEN90
70 INPUT"ARRIVA IL CLIENTE (PER FINIRE FFFF)    ";E$
80 IF E$="FFFF"THEN90
85 GOSUB110:GOTO70
90 IF C$="F"THEN105
100 GOTO10
105 STOP
110 REM IN CODA ***********************
120 IFF=10THENF=1
122 IFF<10THENF=F+1
123 IFF=T THENPRINT"MESSAGGIO DI OVERFLOW":RETURN
125 Q$(F)=E$:IFT=0THENT=1
130 RETURN
140 REM OUT CODA **********************
160 IFT=0THENPRINT"NESSUN ELEMENTO IN CODA":GOTO170
165 E$=Q$(T)
170 IFT=FTHENF=0:T=0:RETURN
175 IFT=0THENT=1
180 IFT<>0THENT=T+1
185 RETURN
210 REM SCANCODA********************
220 PRINT" *** INIZIO CODA ***"
225 FORJ=TTOF:PRINTQ$(J):NEXTJ
230 PRINT" *** FINE CODA ***"
240 RETURN
READY.
```

```
READY.

200 REM ***** SCANCODA *****
205 PRINT"** INIZIO CODA **"
210 IFT<FTHENFORJ=6T TO F:PRINT"CLIENTE";Q$(J):NEXTJ
220 IFF<TTHENFORJ=T TO100:PRINT"CLIENTE";Q$(J):NEXTJ:FORJ=1TO F:PRINT"CLIENTE";Q
$(J):NEXTJ
230 RETURN
READY.
```

I SISTEMI DI NUMERAZIONE

INTRODUZIONE

In questo articolo si discute l'argomento della rappresentazione dei numeri in varie basi, da binaria a esadecimale. Alla fine dell'articolo viene presentato un programmino BASIC che converte un numero rappresentato in una qualunque base compresa tra 2 e 16, in un numero in base dieci.
Il prossimo mese si tratterà della conversione inversa: come rappresentare un numero decimale in una base diversa (binaria, ternaria etc... fino ad esadecimale).
Verrà quindi presentato l'analogo programma BASIC di conversione.

Certamente nessuno dubita che i nomi che gli uomini danno alle cose siano completamente arbitrari. In Francia cane si dice "chien", in Gran Bretagna "dog" e qualunque cosa si dice in cinese è perfettamente chiaro a tutti che ha lo stesso valore di indicare l'animale cane così come in ogni altra lingua. Con i numeri le cose stanno nello stesso identico modo: lo stesso "numero" può essere espresso in diverse "lingue", dove per "lingua" si deve intendere una base di numerazione.
Per la comunicazione tra soli esseri umani, la numerazione adottata è stata la base 10, cioè la più naturale per degli esseri dotati di 10 dita (che si rivelano utilissime per contare).
L'elaboratore al contrario degli uomini, è in grado, assenza o presenza di corrente, di riconoscere solo due diverse cifre: lo Ø e l'1, e questo è un po' come dire che possiede solamente due dita.
Il sistema di numerazione interno di un computer è quindi **"binario"**.
Lo scopo di questo articolo è quello di presentare i diversi possibili sistemi di numerazione usati nel mondo dell'informatica, spiegando anche come passare con facilità dall'uno all'altro.
A questo scopo, ed anche per dar la possibilità di impratichirsi con le varie basi di numerazione, saranno presentati dei programmi BASIC, semplici ma istruttivi, coi quali esercitarsi nelle conversioni.

SISTEMI DI NUMERAZIONE

Noi siamo abituati ad usare i numeri nella rappresentazione decimale; gli elaboratori invece usano normalmente la rappresentazione binaria.
Il sistema decimale ha per "base" il numero 1Ø, in quanto vengono usate 1Ø cifre: Ø, 1, 2, 3, 4, 5, 6, 7, 8 e 9.

Un qualsiasi numero in base 10 è una successione di cifre, ognuna delle quali ha un valore che dipende dalla posizione che la cifra stessa occupa allo interno del numero; il nostro sistema numerico usa cioè una "notazione posizionale". Il numero 442, per esempio, viene interpretato come la somma di 3 numeri, di cui 4 centinaia, 4 decine e 2 unità, cioè

$$442_{base\ 10} = 4 \times 10^2 + 4 \times 10^1 + 2 \times 10^0$$

dove $10^0 = 1$ (un qualunque numero elevato alla potenza zero è uguale ad uno).

Ogni cifra ha un peso diverso a seconda della posizione occupata; abbiamo infatti visto nello esempio precedente che la cifra 4, comparendo in 2 posizioni diverse, assume due pesi diversi: 10^2 e 10.

Il sistema binario, invece, ha per "base" il numero 2, in quanto vengono usate 2 cifre: 0 e 1. Un qualsiasi numero in base 2 è una successione di "zeri" ed "uno", ognuno dei quali, anche in questo sistema, ha un valore che dipende dalla posizione occupata (che si legge "uno zero uno uno").

Il numero binario 1011 può essere così espresso:

$$(1011)_2 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2 + 1 \times 2^0 = 11_{10}$$

Il numero 1011 in "base 2" è quindi uguale ad 11 in "base 10". Ogni cifra del sistema binario è chiamata "bit", nel linguaggio informatico, contrazione di "binary digit" (letteralmente "cifra binaria"); "byte" è invece il nome inglese con il quale si indicano gruppi di 8 bit.

Esistono altri sistemi di numerazione, quali l'"ottale" e l'esadecimale (base 16) usati nel mondo dell'informatica per rappresentare i numeri binari in modo più conciso.

Infatti all'aumentare della base ci sono più simboli a disposizione per rappresentare un numero, ma la lunghezza della rappresentazione diminuisce e il numero diventa più leggibile: per esempio:

$$1111_2 = 15_{10} = F_{16}$$

Il sistema ottale ha come base il numero 8 ed utilizza le cifre 0, 1, 2, 3, 4, 5, 6, 7; il sistema esadecimale ha come base il numero 16 ed utilizza come simboli fondamentali le 10 cifre decimali e le prime 6 lettere dell'alfabeto:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, ed F.

Come potete notare, per i sistemi di numerazione con base maggiore di 10 è necessario introdurne altri simboli, oltre alle cifre numeriche; nel sistema esadecimale sono state utilizzate le prime 6 lettere dell'alfabeto (avrebbero potuto usare altrettanto bene le ultime 6).

decimale	binario	ottale	esadecimale
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10

Tab. 1 - Sistemi di numerazione

Nella tab. 1 potete vedere le cifre fondamentali utilizzate dai 4 sistemi di numerazione considerati e come vengono espressi diversamente i primi 16 numeri nei rispettivi sistemi.
Consideriamo il numero 442 in "base 8", che può essere così scomposto:

$$442_8 = 4 \times 8^2 + 4 \times 8 + 2 \times 8^0 = 98_{10}$$

mentre la scomposizione dello stesso numero in "base 16" risulta:

$$442_{16} = 4 \times 16^2 + 4 \times 16 + 2 \times 16^0 = 690_{10}$$

Il numero 442_8 è quindi diverso dal numero 442_{16}.
Siete ora in grado di convertire un qualunque numero in "base 2, 8 o 16" in un numero in "base 10"; il programma che vi forniamo vi permetterà diesercitarvi e verificare la correttezza delle vostre conversioni.
Nel prossimo numero invece vedremo il metodo inverso: dal sistema numerico decimale a sistemi binari, ottali, esadecimali.
Nei vostri esercizi usate numeri positivi ed interi, in seguito vedremo come rappresentare anche i restanti numeri.

IL PROGRAMMA DI CONVERSIONE

Il semplice programma della pagina successiva, scritto in BASIC COMMODORE 64, vi permette di convertire in decimale, un numero di al massimo 8 cifre espresso in una qualunque base, compresa tra 2 e 16.
Il programma chiede innanzitutto la base di numerazione, e poi chiede le cifre del numero da convertire in decimale.
La logica del programma segue le regole di conversione della numerazione posizionale:

1) dopo aver controllato che il codice ASCII di ogni cifra del numero sia tra quelle permesse all'interno della base dichiarata (istruzioni 70 ÷ 110)...
2) ogni valore della cifra (cioè V) viene moltiplicato per la potenza adeguata della base di numerazione (istruzione 120).

Il programma prosegue a chiedere numero uno dopo l'altro, per fermarlo premere CTL-RESTORE.

```
 10 INPUT "BASE"; B
 20 IF B>10 GOTO 300
 30 INPUT "NUMERO"; A$
 40 IF LEN (A$)>8 THEN PRINT "NON CONVERTO NUMERI
    DI PIÙ DI 8 CIFRE": GOTO 10
 50 TL=0 : C=0
 60 FOR X=LEN(A$) TO 1 STEP —1
 70 C=C+1 : D$=MID$(A$,C,1) : T=ASC(D$)
 80 IF T<48 OR T>70 GOTO 200
 90 IF T>57 AND T<65 GOTO 200
100 IF T>64 THEN V=T—55
105 IF T<58 THEN V=VAL(D$)
110 IF V>B GOTO 200
```

```
120 TL = TL + V * B↑(X—1)
125 NEXT X
130 PRINT "BASE "; B; " NUMERO "; A$
140 PRINT " DECIMALE "; TL
150 GOTO 10
200 PRINT "LA CIFRA "; D$ ; "NON ESISTE NELLA
    NUMERAZIONE IN BASE"; B
205 GOTO 10
300 PRINT "BASI PERMESSE: 2 .... 16"
305 GOTO 10
```

IL BASIC DELLO SPECTRUM 2ª lezione

Nel numero scorso abbiamo affrontato i primi elementi fondamentali della programmazione BASIC; abbiamo visto cosa è una variabile, come sia possibile cambiarne il valore direttamente da programma oppure richiedendolo all'operatore, per finire abbiamo anche esaminato le istruzioni di salto e di stampa.
Avendo ormai assimilato i concetti fondamentali possiamo iniziare in questo numero una trattazione più approfondita delle molte istruzioni che ancora dobbiamo prendere in esame.
Cominciamo col parlare di una struttura fondamentale della programmazione (non solo BASIC): i cicli.
I cicli sono strutture fatte in modo tale che tutto ciò che è compreso all'interno (cioè fra l'istruzione di dichiarazione e quella di conclusione) venga ripetuto un determinato numero di volte. Creare un ciclo in BASIC utilizzando solo le istruzioni che abbiamo visto finora è impossibile in quanto occorre eseguire un test a fine ciclo per verificare che si siano o meno raggiunte le condizioni richieste nell'istruzione di dichiarazione: noi non abbiamo ancora esaminato queste istruzioni ma lo faremo in seguito. Per il momento vedremo quindi solo il metodo più semplice per realizzare cicli in BASIC. Un programma che produca un ciclo deve possedere, come già detto, due istruzioni che controllano lo stesso, una all'inizio per determinare il numero di volte che il ciclo deve essere ripetuto e una alla fine per incrementare la variabile contenente il numero di cicli già svolti e, se questo è minore di quelli richiesti all'inizio, eseguire un nuovo ciclo.

```
10 FOR A = 1 TO 10000
20 PRINT A
30 NEXT A
```

Il programmino dell'esempio esegue ben 10000 cicli. Le istruzioni delle linee 10 e 30 sono quelle di controllo mentre tutto ciò che si trova all'interno (Linea 20) sarà eseguito per 10000 volte.
L'istruzione della Linea 10 in italiano significa - per A = 1 fino B 10000 -. Questo vuol dire che viene dichiarata una variabile destinata al controllo di un ciclo la quale assume come valore iniziale i e deve arrivare fino a 10000 per un totale quindi di 10000 ripetizioni. L'istruzione di controllo alla linea 30 incrementa il contenuto della variabile A e se questa risulta minore di 10000 esegue un nuovo ciclo saltando alla prima linea interna al ciclo (in questo caso la 20).
Naturalmente tutte le istruzioni interne a un ciclo non devono alterare il valore della variabile di controllo altrimenti il ciclo risulterebbe falsato o peggio po-

trebbe entrare in un giro continuo senza mai smettere. Nello esempio precedente abbiamo posto all'interno del ciclo un'istruzione PRINT A la quale stampa il contenuto della variabile di controllo mostrando continuamente quante ripetizioni sono state compiute fino a quel momento.
In alcune circostanze potrà succedere che dobbiate sviluppare dei cicli interni ad altri (cicli annidati), ciò è possibile usando due diverse variabili di controllo e operando nel modo seguente.

```
10 FOR A=1 TO 100
20 FOR B=1 TO 500
30 NEXT B
40 NEXT A
```

In questo modo avrete il ciclo B interno al ciclo A quindi per ogni ciclo di A vengono eseguiti 500 cicli di B; cercate di capirne il motivo ricordando quello che vi abbiamo detto in precedenza e cioè che l'istruzione NEXT causa un salto alla prima istruzione interna al ciclo.
Il programma dell'esempio genera solamente dei cicli vuoti quindi non ha alcun effetto se non quello di creare delle pause (ci vuole infatti del tempo per eseguire tante volte tutte queste istruzioni) ma se noi inserissimo delle istruzioni fra le linee 20 e 30 esse sarebbero ripetute ben 50000 volte, questo a causa di ciò che abbiamo detto prima da cui si ricava che eseguendo 500 cicli interni per ogni ciclo esterno e essendo questi 100 si ha:

```
500*100 = 50000
```

Fino ad adesso abbiamo supposto che l'incremento della variabile di controllo fosse di una unità alla volta e ciò è vero a meno che non venga dichiarato un fattore di incremento diverso. L'istruzione preposta a identificare il ritmo di incremento è STEP (passo) e deve trovarsi nella linea di dichiarazione, come segue.

```
10 FOR A=1 TO 100 STEP 2
20 PRINT A
30 NEXT A
```

Questo programma esegue un ciclo da 1 a 100 con un passo di 2 quindi, partendo da 1, la variabile conterrà successivamente 3, 5, 7, 9...
Procedendo con un passo maggiore il ciclo non sarà più ripetuto 100 volte ma, in questo caso, 50 e stamperà sullo schermo tutti i numeri dispari da 1 a 99, questo perché i numeri pari vengono saltati. Per modificare il programma in modo che stampi solo i numeri pari si può procedere nel modo seguente.

```
10 FOR A=2 100 STEP 2
20 PRINT A
30 NEXT A
```

In questo modo la variabile di controllo viene inizializzata a 2 anziché a 1 e quindi un incremento di 2 unità salterà tutti i numeri dispari.
Naturalmente è anche possibile stabilire dei passi negativi nel qual caso il valore iniziale della variabile di controllo dovrà essere superiore rispetto al valore limite da raggiungere,

ecco un esempio.

```
10 FOR A = 500 TO 1 STEP -1
20 NEXT A
```

La clausola STEP nella dichiarazione di un ciclo non è obbligatoria e se omessa viene assunta uguale a 1.
In precedenza abbiamo detto che è possibile creare un ciclo anche senza ricorrere alla struttura FOR...NEXT a condizione però di poter controllare se si è verificata la condizione di fine ciclo e agire di conseguenza.
Le strutture decisionali in grado di reagire in modo diverso a diverse situazioni sono importantissime e indispensabili, è impensabile sviluppare dei programmi senza il loro ausilio.
In BASIC l'istruzione di verifica è la seguente:

```
IF (condizione)
THEN (istruzione)
```

La condizione successiva a IF può essere un qualsiasi confronto tra 2 valori usati nell'ambito del programma. Per esempio.

```
IF A > 10 THEN PRINT "CIAO"
```

Significa:

```
SE A > 10 ALLORA STAMPA
"CIAO"
```

Per il confronto tra due grandezze si possono usare i seguenti simboli:

= uguale
< minore
> maggiore
< = minore o uguale
> = maggiore o uguale
< > diverso

Per chiarire le idee, se A = 20 e B = 100 la condizione A = B è chiaramente falsa e non avrà quindi effetto, ponendo invece una condizione del tipo A < B si può facilmente capire come questa sia vera infatti il contenuto della variabile A (20) è indubbiamente minore di quello della variabile B (100). Vediamo un semplice programmino che utilizzi questa istruzione.

```
10 INPUT "PRIMO NUMERO?
   ";A
20 INPUT "SECONDO
   NUMERO? ";B
30 IF A < B THEN PRINT A;
   "<";B
40 IF A = B THEN PRINT A;
   " = ";B
50 IF A > B THEN PRINT A;
   ">";B
60 INPUT "SI PER
   CONTINUARE";C$
70 IF C$ = "SI" THEN GOTO
   10
80 STOP
```

Alle linee 10 e 20 vengono richiesti i valori da assegnare alle variabili A e B; la linea 30 controlla se A < B nel qual caso stampa il valore di A e di B legati dal simbolo < (vedi puntata precedente sulla PRINT), la linea 40 controlla se i due valori sono uguali e agisce di conseguenza mentre la linea 50 verifica il caso in cui A maggiore di B. Chiaramente il verificarsi di una condizione esclude automaticamente le altre due, infatti è impossibile che un numero sia contemporaneamente maggiore e uguale all'altro. Finiti i

test la linea 60 richiede di rispondere SI alla richiesta di input (stavolta alfanumerico) se si desidera utilizzare ancora il programma; alla linea successiva troviamo un'altra verifica per controllare se la stringa inserita (C$) corrisponde alla parola SI e in caso positivo salta alla linea 10 da cui il programma riparte.
Come abbiamo visto si possono eseguire anche comparazioni alfanumeriche oltre che numeriche, ma non è ovviamente possibile confrontare numeri con stringhe o viceversa.
In molti casi perché sia eseguita una istruzione occorre che si verifichino anche 2 o più condizioni contemporaneamente il che risulta abbastanza scomodo da fare sapendo solo quello che abbiamo detto finora, ecco allora che ci vengono in aiuto altre due utilissime istruzioni complementari: AND e OR.
Prima di spiegare a cosa servono queste due nuove istruzioni proviamo a realizzare un programma che esegua un comando solo al verificarsi di due condizioni simultaneamente.

```
10 INPUT "PRIMO ";A
20 INPUT "SECONDO ";B
30 INPUT "TERZO ";C
40 IF A>B THEN GOTO 50
45 GOTO 10
50 IF B>C THEN PRINT
   "GIUSTO"
60 STOP
```

Le prime linee di programma richiedono all'utente tre valori da assegnare a altrettante variabili denominate A, B e C. Perché la verifica abbia esito positivo il valore di A deve essere maggiore di B il quale a sua volta deve essere maggiore di C. Vediamo quindi che la linea 40 esegue un test per la prima condizione e se questa risulta soddisfatta salta alla linea 50 dove viene testata la seconda condizione, se anche questa è verà sarà stampato GIUSTO, in caso contrario il programma si arresta, ma notate che c'è anche una linea 45 la quale causa un salto alla fine del programma nel caso in cui la prima condizione non sia vera, quindi perché il programma dia come risultato GIUSTO bisogna avere inserito i tre numeri con valori decrescenti (es: 30, 20, 10). Questo tipo di verifica si chiama AND ed è necessario che tutte le condizioni siano vere per ottenere un risultato positivo. Il programma equivalente a quello sopra descritto diverrà ora:

```
10 INPUT "PRIMO ";A
20 INPUT "SECONDO ";B
30 INPUT "TERZO ";C
40 IF A>B AND B>C THEN
   PRINT "GIUSTO"
50 STOP
```

Come potete vedere le due condizioni da verificare vengono legate dell'istruzione AND all'interno della stessa IF risparmiando molto lavoro inutile.
Non sempre però è necessario che in una serie di condizioni tutte debbano necessariamente verificarsi, per esempio su 3 verifiche potrebbe bastare che solo una sia vera per determinare l'esecuzione dell'istruzio-

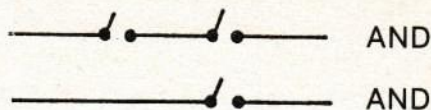
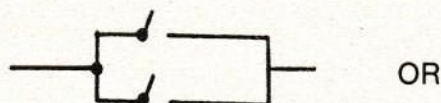
ne. Vediamo come procedere per un simile test riferendoci agli esempi precedenti.

```
10 INPUT "PRIMO ";A
20 INPUT "SECONDO ";B
30 INPUT "TERZO ";C
40 IF A>B THEN GOTO 80
50 IF B>C THEN GOTO 80
60 GOTO 90
80 PRINT "GIUSTO"
90 STOP
```

In questo esempio basta che una sola delle due condizioni si verifichi, infatti dopo la richiesta dei tre numeri viene eseguito un test per controllare se A>B, in caso positivo avviene un salto alla linea 80 la quale stampa GIUSTO, se la prima condizione è falsa si controlla subito la seconda condizione: se anche questa è falsa viene eseguita l'istruzione successiva che causa un salto alla fine del programma (STOP), se invece la condizione è vera avviene come prima un salto alla linea 80.

Questo sistema di concatenazione fra due o più condizioni si chiama OR. Vediamo il programma equivalente.

```
10 INPUT "PRIMO ";A
20 INPUT "SECONDO ";B
30 INPUT "TERZO ";C
40 IF A>B OR B>C THEN
   PRINT "GIUSTO"
50 STOP
```

Ripetiamo per l'ennesima volta che usando il comando OR è sufficiente che una delle condizioni si verifichi mentre con AND tutte le condizioni devono risultare vere. Paragonando queste istruzioni a un sistema elettrico possiamo ricavare la seguente analogia.

Come potete vedere anche se una delle due condizioni risulta vera (interruttore chiuso) rimane l'altra a impedire il passaggio.

In questo caso è sufficiente che una delle due condizioni si verifichi per consentire il passaggio. Solo nel caso in cui entrambe siano false il test non avrà successo.

Per concludere vediamo come simulare una struttura del tipo FOR...NEXT usando al loro posto queste nuove istruzioni.

```
10 LET A=1:LET B=1000
20 PRINT A
30 LET A=A+1:IF A<=B
   THEN GOTO 20
```

Come possiamo facilmente notare, nella prima linea (quella di dichiarazione) occorre definire 2 variabili, una contiene il valore iniziale che sarà incrementa-

to a ogni ciclo mentre l'altra deve contenere il valore limite, raggiunto il quale il ciclo termina; le linee fra la 10 e la 30 contengono le istruzioni da eseguire (in questo caso c'è solo la 20).
La linea 30 corrisponde a un NEXT ma è decisamente più lunga e scomoda in quanto composta di ben tre istruzioni: la prima incrementa di 1 (il passo è sottinteso) la variabile di controllo A, la seconda istruzione controlla se il nuovo valore di A è minore o uguale a quello della variabile limite B e, in caso positivo salta alla prima linea interna al ciclo (20). Come vedete alla linea 30 sono state messe 2 istruzioni sulla stessa linea separandole con i due punti, proprio come avevamo visto nella scorsa puntata a proposito delle PRINT. A questo riguardo precisiamo che anche dopo un THEN è possibile porre diverse istruzioni separandole con i due punti ma esse verranno eseguire solo se la condizione è verificata.
Ora che abbiamo imparato a usare queste importanti istruzioni possiamo passare a esaminare un altro tipo di variabili che non abbiamo preso in esame la volta scorsa per motivi pratici.
In realtà non si tratta di un diverso tipo di variabile ma di un diverso modo di utilizzare le variabili in maniera molto pratica e funzionale. Si tratta delle variabili indicizzate.
Le normali variabili, come certo ricorderete, sono solo delle lettere che rappresentano un valore numerico; l'uso di queste variabili va benissimo per la maggior parte delle applicazioni che ne prevedono un uso limitato e non ordinato ma per certi scopi è impensabile l'uso di moltissime variabili con un nome diverso. Per esempio, per confrontare un valore con quello contenuto in 100 variabili occorrerebbe necessariamente eseguire 100 confronti diversi e sarebbe ben difficile individuare le variabili contenenti quel valore.
A questo punto ci vengono in aiuto le variabili indicizzate composte da una o più lettere che distinguono la matrice di appartenenza, più un numero posto fra parentesi che indica l'elemento del vettore in esame. Questo tipo di variabili deve essere preventivamente dichiarato con l'istruzione DIM.
Per esempio, se vogliamo creare una matrice di nome A composta da 100 variabili numeriche useremo l'istruzione:

```
DIM A (100)
```

In questo modo definiamo un vettore di 100 elementi con lo stesso nome (A) selezionabili variando l'indice posto fra parentesi. Per chiarire le idee, la prima variabile sarà A (0), poi a (1), A (2), ecc.
Per assegnare un valore a una variabile di questo tipo, si procede nel solito modo visto la volta scorsa.
Vediamo un programma di esempio che crea un vettore di 20 elementi e assegna a ognuno di essi il valore 5.

```
10 DIM U (19)
20 FOR A = 0 TO 19
25 LET U(A) = 5
30 NEXT A
40 STOP
```

La linea 10 dimensiona il vettore numerico di 20 elementi (da 0 a 19) chiamato U; la linea 20 inizia un ciclo che varia appunto da 0 a 19 all'interno del quale una istruzione di assegnazione usa la variabile di controllo del ciclo come puntatore per gli elementi del vettore che vengono "ripescati" uno alla volta e forzati a contenere il valore 5. Come potete osservare l'unica cosa che distingue queste variabili indicizzate è appunto l'indice che punta all'elemento interessato del vettore. Naturalmente è possibile dimensionare anche variabili stringa semplicemente rispettando la solita sintassi che richiede il simbolo $ come ultimo carattere nei nomi delle variabili stringa. Per riassumere il tutto vediamo un esempio di programma per la rubrica telefonica che, dopo aver dimensionato due vettori alfanumerici di 10 elementi ciascuno richiede il contenuto da assegnare a ogni elemento considerando che la prima richiesta riguarda il nome mentre la seconda il numero telefonico.

```
10 DIM N$ (10): DIM T$ (10)
15 FOR A = 1 TO 10
20 INPUT "NOME? ";N$(A)
25 INPUT "TEL.? ";T$(A)
30 NEXT A
35 CLS
40 FOR A = 1 TO 10
45 PRINT "NOME ";N$(A)
50 PRINT "TEL. ";T$(A)
55 PRINT
60 NEXT A
70 STOP
```

Come già detto, all'inizio vengono creati due vettori chiamati N$ e T$. Le linee successive chiedono all'utente il nome e il numero telefonico da assegnare alla variabile puntata dalla variabile di controllo A. Il tutto viene quindi ripetuto per 10 volte. Terminata l'operazione di caricamento dei dati il programma stampa ordinatamente tutti i nomi e numeri inseriti tramite il ciclo che va dalle linee 40 a 60; anche qui per cambiare il valore dell'indice si usa la variabile di controllo del ciclo. La linea 35 serve a cancellare lo schermo, ma questo per il momento non ci interessa.
Il programmino usato come esempio è una versione molto semplice e di scarsa utilità (alla fine dell'articolo troverete una versione molto migliore) ma provate a farlo senza usare le variabili indicizzate!
Bene, adesso sappiamo che le variabili indicizzate rappresentano una lunga fila di variabili distinte dal numero fra parentesi e sappiamo anche che il numero di variabili che abbiamo intenzione di usare deve essere dichiarato nell'istruzione DIM; questa istruzione però permette anche di definire matrici di variabili multidimensionali, composte cioè da più vettori. Chiariamo subito con un esempio: quello che segue è un sem-

plice vettore monodimensionale di 4 elementi creato con l'istruzione DIM A(4).

A(1), A(2), A(3), A(4)

Vediamo invece come è composta una matrice a 2 dimensioni con il primo vettore di valore 4 (4 elementi) a cui è associato il secondo di valore 2 (2 elementi) creato con una istruzione DIM A(4,2).

A(1,1);A(2,1);A(3,1);A(4,1)
A(1,2);A(2,2);A(3,2);A(4,2)

Vi sarete già accorti che sia nell'istruzione di dimensionamento (DIM) che nelle variabili vi sono due numeri fra parentesi che corrispondono agli indici dei due vettori. In questo caso il primo vettore è composto di 4 elementi mentre il secondo di 2 quindi in totale disporremo di 10 variabili (5*2).
Questo tipo di matrici è molto utile in diverse applicazioni, vediamo un esempio del loro utilizzo nel programma che segue, il quale crea una tabellina pitagorica e quindi utilizza una matrice di 10*10.

```
10 DIM N(10,10)
15 FOR A=1 TO 10
20 FOR B=1 TO 10
25 N(A,B)=A*B
30 NEXT B
35 NEXT A
```

A questo punto se scriverete:

```
PRINT N(5,8)
```

Otterrete la stampa di 40, infatti 5*8=40. Lo stesso avverrà con qualsiasi combinazione di numeri sceglierete purché non superi il limite massimo di 10*10 ma vediamo come siamo riusciti a ottenere tutto questo.
Le linee 15 e 20 formano due cicli uno interno all'altro e in mezzo a questi due si trova l'istruzione di assegnazione dove la variabile di controllo del ciclo esterno è usata come puntatore della prima dimensione mentre la variabile del ciclo interno punta la seconda dimensione e noi sappiamo che per ogni ciclo esterno ne vengono eseguiti 10 interni che corrispondono appunto alle "righe" di una ipotetica tabellina pitagorica; il valore da assegnare alla variabile interessata viene ottenuto semplicemente moltiplicando le due variabili di controllo A e B. Potrete capire meglio il tutto osservando lo schema riportato più avanti.
Per concludere, un accenno sul modo di cancellare le variabili presenti in memoria; ciò è possibile con il comando CLEAR, ma fate molta attenzione a usarlo. Se oltre alle variabili si desidera cancellare anche tutto il programma occorre il comando NEW, potete usarlo ora prima di digitare il programma che segue. Arrivederci...

2 continua

Massimo Cellini

LE PERIFERICHE: DI INGRESSO E USCITA

Dispositivo	Ingresso	Uscita
lettore di schede	x	
perforatore di schede		x
lettore di nastro perforato	x	
perforatore di nastro		x
stampante		x
terminale	x	x
console	x	x
video		x
unità a nastro magnetico	x	x
unità a disco magnetico	x	x
lettore ottico	x	
lettore di caratteri magnetici	x	

LA CONSOLLE

La consolle è il «quadro di comando» dell'intero sistema e dell'unità centrale in particolare. È composta da:
— un'unità dattilografica da cui si inviano i comandi e le istruzioni particolari;
— un'unità video in cui si possono vedere i momenti di una elaborazione e le informazioni che l'unità centrale può trasmettere per avere ulteriori istruzioni o informare l'operatore sulle fasi particolari dell'elaborazione;
— un pannello luminoso composto di spie che permettono agevolmente di controllare lo svolgimento del processo, intervenendo, se del caso, con opportuni comandi.
È collegata con la stampante e ogni altra unità periferica.

IL TERMINALE

È un supporto temporaneo di informazione, spesso costituito:
— da una tastiera (tipo macchina per scrivere);
— da uno schermo visivo (genere schermo televisivo o tubo catodico);
— ed eventualmente da un sistema stampante che dà origine a un supporto di carta permanente.
Ma vi sono altri sistemi, quali:
— i dispositivi emittenti d'informazione: lettori automatici di schede perforate, di etichette, di segni magnetici; segnalatori automatici di vario tipo (adattati per esempio alla pesa automatica, a contatori, ecc);
— i dispositivi riceventi: cuffie telefoniche le quali consentono l'ascolto di informazione verba-

le; dispositivi che selezionano una veduta di microfilm proiettata su uno schermo.
Tutti questi dispositivi sono caratterizzati dalla non-permanenza dell'informazione che viene manipolata; essi sono per natura destinati a ritornare allo stato neutro in un tempo più o meno breve.

L'UNITA' VIDEO

Le unità di visualizzazione (video-terminali) permettono un dialogo diretto (domanda-risposta) con il calcolatore. Le comunicazioni del calcolatore appaiono su di uno schermo televisivo.
Le domande vengono fatte per mezzo di una tastiera, di cui è dotata l'unità video. Lo schermo può contenere sino a 24 righe di 80 posizioni l'una.

IL LETTORE OTTICO DI DOCUMENTI

È in grado di riconoscere i caratteri alfanumerici scritti in stampatello ed i caratteri speciali. Una sorgente luminosa emette un fascio di luce che colpisce il documento, suddiviso in tanti quadratini che formano un reticolo. Ogni quadratino viene controllato per determinare il suo contenuto di NERO o BIANCO. Il reticolo viene successivamente confrontato con i caratteri standard; in caso di eguaglianza si ha la conversione in forma digitale e il segnale viene inviato all'unità centrale.

LA STAMPANTE

La stampante è un'unità di uscita ed appartiene ai dispositivi stampanti. Questi sono organi di uscita specializzati, i quali forniscono documenti direttamente utilizzabili dall'uomo. Si possono distinguere:
1) Le stampanti lente: incorporate nella consolle dell'elaboratore, queste macchine sono a battuta elettrica. Comandata direttamente dall'unità centrale alla velocitá stampante, che va da 10 a 20 caratteri per secondo (i famigerati "C e S"), la macchina da scrivere può:
— sia ricevere messaggi dall'unitá centrale (o immetterne, essendo reversibile, e in tal caso agisce come tastiera);
— sia negli elaboratori di piccola potenza, assicurare la stampa dei risultati, quando tali risultati sono poco numerosi.
2) Le stampanti rapide.
I caratteri stampanti, montati su un cilindro o su una catena, hanno un movimento di rotazione permanente. La battuta si fa in volata, cioè senza arresto dei caratteri stampanti: un martelletto applica il carattere voluto, durante un tempo molto breve, sulla carta dove si stampa il testo, nel momento in cui il carattere passa davanti alla carta.
La macchina assicura la stampa simultanea di tutti i caratteri presenti nella riga da stampare (la capacitá va oggi da 120 a 160 caratteri per riga). La velocità viene espressa nel numero di righe stampate per minuto (da 200 a 1500 righe per minuto). Come si vede, si tratta di macchine a grande portata, corrispondente alla rapidità di cadenza dell'unitá centrale. Ma

la stampa rimane ancora come una strozzatura della periferia rispetto all'unità centrale. Attualmente si cerca di accrescere la velocità di stampa (stampante in cui la velocità è portata a 10.000 righe per minuto, funzionante a raggi laser).

IL TRACCIATORE GRAFICO

È un'unità di uscita. È costituito da un tamburo rotante sul quale scorre un foglio di carta millimetrata ed un «penna» montata su un carrello di modo che il suo moto sia orizzontale lungo la larghezza del tamburo. È possibile ottenere qualsiasi movimento dell'organo scrivente con solo 6 comandi:

destra	sinistra
verso l'alto	verso il basso
penna in su	penna in giù

Si possono ottenere linee rette e curve.

LE SCHEDE PERFORATE

La scheda perforata è stata il supporto più usato per l'introduzione di informazioni in un calcolatore. La scheda è un cartoncino rigido su cui, mediante opportune perforazioni, vengono registrate le informazioni, costituite da un insieme di caratteri alfanumerici e di caratteri speciali.

Benchè non si possa parlare di un formato standard, le schede più diffuse hanno le seguenti dimensioni (in mm): altezza 82,5, larghezza 187,3 e spessore 0,2. La scheda è organizzata in colonne e righe: precisamente 80 colonne, numerate sulla scheda da 1 a 80, e 12 righe. Queste ultime sono indicate solo da 0 a 9, mentre sulla fascia superiore compaiono altre due righe non numerate, dette convenzionalmente 11 e 12.

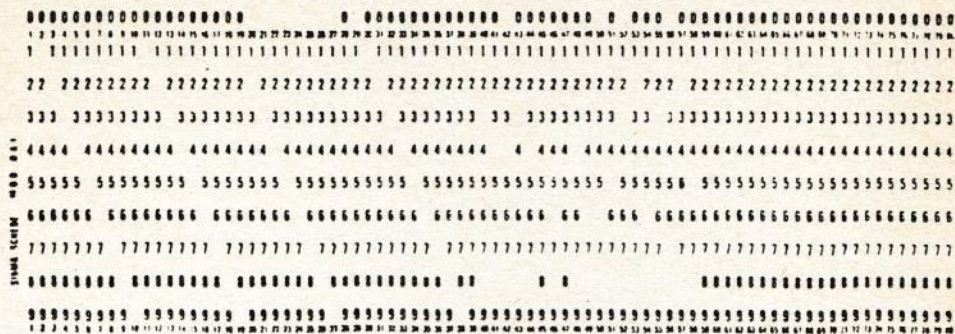
Su ogni colonna trova posto un carattere, rappresentato mediante perforazioni in una o più righe. Su ogni scheda quindi possono essere registrati 80 caratteri, uno per ogni colonna.

La corrispondenza fra carattere e perforazioni è stabilita da un codice: uno dei più noti è il codice Hollerith.

carattere	perforazione	carattere	perforazione	carattere	perforazione
A	12-1	P	11-7	4	4
B	12-2	Q	11-8	5	5
C	12-3	R	11-9	6	6
D	12-4	S	0-2	7	7
E	12-5	T	0-3	8	8
F	12-6	U	0-4	9	9
G	12-7	V	0-5	(	12-8-5
H	12-8	W	0-6	)	11-8-5
I	12-9	X	0-7	.	12-8-3
J	11-1	Y	0-8	,	0-8-3
K	11-2	Z	0-9	;	11-8-7
L	11-3	0	0	/	0-1
M	11-4	1	1	=	0-8-5
N	11-5	2	2		8-6
O	11-6	3	3		12-8-6

ABCDEFGHIJKLMNOPQRSTUVWXYZ 0123456789 ¢@%$*&+-<=>(,:;.?!)

riga 12

riga 11

PERFORATRICE DI SCHEDE

Le macchine con cui vengono perforate le schede vengono dette perforatrici e concettualmente non sono dissimili da una comune macchina da scrivere. La carta è sostituita da un magazzino di schede ed i martelletti con i caratteri da punzoni. Opportuni comandi prelevano le schede ad una ad una dal magazzino e le passano sotto la stazione di perforazione che contiene 12 punzoni, uno per ogni riga. Quando si batte un tasto scattano i punzoni opportuni che effettuano le perforazioni corrispondenti al carattere battuto, secondo il codice usato dalla perforatrice. La scheda quindi avanza di una colonna ed è pronta per un'altra perforazione.
La perforatrice, non avendo alcun collegamento diretto con il calcolatore, non è un organo di ingresso/uscita, ma solo un dispositivo ausiliario.

IL PERFORATORE DI SCHEDE

Questo dispositivo ha la funzione di perforare su schede i dati inviati in uscita dal calcolatore. Questo processo può avvenire in modo seriale e parallelo, con tecniche sostanzialmente identiche a quelle usate per i lettori di schede. Cambiano evidentemente le testine che sono sostituite da testine di perforazione, e cambia il verso delle informazione che, questa volta, va dall'elaboratore al dispositivo di uscita.
Molto spesso le funzioni di lettura e perforazione vengono svolte da una sola unità di ingresso/uscita detta lettore/perforatore di schede.

IL LETTORE DI SCHEDE

Il lettore di schede è una unità d'ingresso che opera sotto il controllo del governo del calcolatore. Ha la funzione di raccogliere le informazioni, contenute su schede perforate, e di inviarle alla memoria. I metodi di lettura di una scheda sono due: seriale e parallelo.
metodo seriale: la scheda, in questo caso, avanza, colonna per colonna, sotto il dispositivo

di lettura. I caratteri vengono letti serialmente, uno per volta. I dati possono essere trasmessi alla Memoria uno per volta o può esistere una memoria di transito, comunemente detta buffer, che li raccoglie prima di inviarli alla memoria centrale. In questo ultimo caso la trasmissione avviene solo quando è stata completata la lettura di una intera scheda.
Le testine di lettura possono essere sia a spazzolini metallici che a cellule fotoelettriche. La stazione di lettura a spazzolini è costituita da 12 spazzolini di materiale ad alta conducibilità elettrica che strisciano su un rullo, metallico. In fase di lettura la scheda si interpone fra gli spazzolini ed il rullo interrompendo il contatto elettrico. Quando uno spazzolino incontra una perforazione, andando a contatto con il rullo, chiude il circuito elettrico.
Nelle testine a cellule una serie di 12 lampadine è contrapposta a 12 cellule fotoelettriche. La scheda, passando in mezzo, interrompe o meno il flusso luminoso, provocando, di conseguenza, degli impulsi elettrici ai capi delle cellule.
Metodo parallelo: la scheda, in questo caso, viene letta riga per riga e la lettura si conclude quindi in 12 passi anzichè in 80: da qui la maggiore velocità di lettura del metodo parallelo rispetto a quello seriale. È evidente che i caratteri contenuti su una scheda sono, però, decifrabili solo a fine lettura, in quanto sono registrati colonna per colonna. Di qui la necessità di dotare questi lettori di un buffer che contiene una immagine delle informazioni contenute nella scheda. I dati contenuti in questa memoria di transito vengono presi serialmente, cioè colonna per colonna, ed inviati al calcolatore. Un lettore di tipo seriale può leggere al più 400 schede al minuto, mentre un lettore parallelo legge anche 1000 schede al minuto.

IL NASTRO PERFORATO.

Lettore/perforatore di nastro.

La differenza sostanziale fra nastro e schede è che il nastro è un mezzo di registrazione continuo, mentre le schede hanno una lunghezza fissa (80 caratteri).
Anche nel caso del nastro di carta i dati sono registrati come perforazioni su colonne, cioè in senso trasversale alla lunghezza del nastro. Le perforazioni sono, di solito, rotonde: il loro numero varia a seconda del carattere, del codice usato, del tipo di nastro.
Il numero di righe varia da nastro a nastro ed è, in genere, variabile da cinque a otto. Il numero dei caratteri rappresentabili aumenta con il numero di righe del nastro utilizzato. In genere è presente una ulteriore riga, con perforazioni un po' più piccole e continue, che ha funzione di guida per il nastro.
Generalmente è presente una riga di CHECK che ha la funzione di controllo: essa infatti viene perforata o no, a seconda che il numero di perforazioni in quella colonna sia pari o dispari. Abitualmente si dice che questa riga ha funzione di controllo della disparità. La verifica del codice vien fatta dal calcolatore e l'accordo garantisce

entro certi limiti, l'esattezza del dato ricevuto.
Il numero di caratteri che è possibile registrare su un nastro di carta è, ovviamente, uguale al numero di combinazioni possibili delle perforazioni (256 nel caso di nastro ad 8 piste e 32 per uno a 5 piste).
La lettura di un nastro di carta può essere fatta solo in modo seriale, a causa della sua lunghezza non ben definita. Sono disponibili testine di lettura del tipo a spazzole e del tipo a fotocellule. Nel primo caso la velocità di lettura è di circa 180 caratteri al minuto, nel secondo può raggiungere i 1000 caratteri.
Anche la perforazione può essere fatta solo in modo seriale. La stazione di perforazione è simile a quella di un perforatore di schede, a parte la forma dei punzoni e la forza ad essi impressa: il nastro di carta è molto più leggero e meno resistente delle schede. Molto spesso il lettore e il perforatore sono accoppiati in un unica unità.

NASTRO MAGNETICO; Unità a nastro magnetico.

Il nastro magnetico che si usa per gli elaboratori di dati è simile a quello utilizzato dai comuni registratori: cambiano le dimensioni, la qualità, il metodo di registrazione. Analogamente a quanto si fa con il nastro di carta, le cifre binarie, che rappresentano un carattere, vengono registrate lungo la stessa riga trasversale del nastro, su otto piste diverse. La nona pista, come la riga check per il nastro di carta, ha la funzione di controllo di parità. (Nella nona riga viene registrato un 1, se il numero di 1 nelle altre piste è pari; uno zero, se il numero di uno nelle altre piste è dispari).
I dati vengono registrati su nastro magnetico uno di seguito all'altro e la fine di una registrazione è indicata da uno spazio vuoto, lungo circa 0.6 pollici. La funzione di questo spazio vuoto è quella di permettere al meccanismo di trascinamento del nastro di fermarsi a fine lettura e di riprendere la velocità di regime, prima della lettura successiva.
Questo risulta necessario giacchè la lettura o la registrazione dei dati è fedele solo se il nastro si muove alla velocità standard prevista. L'insieme delle informazioni contenute in una registrazione è detto record: più precisamente, viene detto record fisico l'insieme di dati contenuti fra due spazi vuoti successivi.
Il calcolatore è strutturato per leggere record fisici e, al comando di lettura, trasferisce dal supporto in memoria centrale un intero record fisico.
Il nastro magnetico consente di leggere i records sequenzialmente, non è conveniente fare salti alla ricerca di una informazione, poichè i tempi di ricerca sul nastro sono molto alti. Generalmente si utilizzano nastri lunghi circa 700 metri e si registrano con una densità fino a 1600 caratteri per pollice. La velocitá di lettura/scrittura arriva sino a 300.000 caratteri per secondo.

DISCO MAGNETICO; Unità a disco magnetico.

Il disco magnetico è costituito da un piatto circolare di alluminio sulle cui facce è applicato uno strato di materiale magnetizzabile. Le piste sono circolari e concentriche, invece che a spirale continua come nei comuni dischi: ciò è dovuto al fatto che, mentre in questi ultimi è il solco a guidare il braccio, nei dischi del calcolatore il braccio deve potersi muovere liberamente, per assicurare l'accesso diretto alla pista o a parte di essa. Abbiamo detto, riguardo al nastro, che non consente di accedere direttamente a particolari informazioni ma è necessario trattarle sequenzialmente. Nel disco, pur essendo assicurati i vantaggi della registrazione magnetica, è possibile accedere ai dati direttamente (modo direct o random).
I bits in cui vengono codificati i caratteri sono registrati sequenzialmente sulla medesima pista. Il disco è tenuto costantemente in rotazione e l'accesso alle piste è realizzato spostando il braccio recante la testina trasversalmente. In realtà esistono due bracci distinti e solidali tra loro, uno per la faccia superiore e l'altro per la faccia inferiore del disco. Il meccanismo di spostamento del braccio è elettromagnetico o pneumatico. Lo spostamento massimo del braccio, cioè dalla pista più interna a quella più esterna, avviene in qualche decina di millesimi di secondo.
I sistemi di elaborazione di grandi dimensioni utilizzano unità a dischi con più piatti sovrapposti. L'insieme delle piste corrispondenti sui vari dischi viene chiamata cilindro e l'insieme rigido dei piatti è detto DISK-PACK.
La velocità di rotazione del disco è di circa 4000 giri al minuto. La velocità di lettura/scrittura si aggira sui 300.000 caratteri per secondo.

ISTRUZIONI PER LA CASSETTA COMPUTING VIDEOTECA N. 4

COME INIZIARE

(Lato A: ZX Spectrum)

Come ben sapete, COMPUTING VIDEOTECA vuole offrire il meglio ai suoi lettori, presentando programmi e articoli utili e divertenti.

Purtroppo i tempi di caricamento dei programmi da nastro sono piuttosto lunghi e spesso si è costretti a passare diversi minuti davanti allo schermo muto. Da ora in poi ciò non avverrà più, almeno per i programmi di COMPUTING, abbiamo infatti messo a punto un efficiente sistema di memorizzazione dei programmi su nastro che riduce drasticamente i tempi di caricamento, sicuri di fare cosa gradita alle migliaia di appassionati che fedelmente ci seguono.

ISTRUZIONI DI CARICAMENTO

Per caricare i programmi inserite la cassetta nel registratore e scrivete LOAD "" (le " si ottengono con SYMBOL SHIFT e P), subito dopo premete il tasto ENTER e avviate il registratore in PLAY.

A questo punto basterà seguire le istruzioni che appariranno sul video.

Quando avrete finito di usare un programma spegnete il computer staccando per un momento il cavetto di alimentazione. Battete di nuovo LOAD "" seguito da ENTER per caricare il programma successivo.

MIXXIL

Sei stato nominato Comandante di una Base di Difesa Terrestre, il cui compito è di proteggere lo spazio territoriale assegnato dal Comando Strategico.

La zona da difendere è colorata in verde sul fondo del tuo schermo, la crocetta nel centro è la tua base.

Lo scopo del gioco è di non fare toccare terra alle forze di invasione, che compariranno sullo schermo sotto forma di quadrati neri numerati.

Lo spazio aereo da te controllato è idealmente diviso in settori di coordinate X da 1 a 32 (larghezza dello schermo) e Y da 1 a 20 (altezza dello schermo).

Es.: l'angolo superiore sinistro sarà il settore 1,1
l'angolo superiore destro sarà il settore 32,1

Le armi a tua disposizione sono:

1) 20 Missili, numerati da 1 a 20
2) Una postazione Laser

MISSILI:

Per lanciare un missile bisogna, prima di tutto, armarlo, poi dargli l'angolazione di volo.

Per armare un missile premi il tasto "d".

Il missile è pronto al decollo, attende solo l'angolo di volo che gli vuoi assegnare.

Considera la crocetta della tua Base come una Rosa dei Venti in cui l'estrema sinistra è l'OVEST e l'estrema destra l'EST, la linea che collega questi due punti cardinali è un angolo "piatto" di 180 gradi dove EST è a 0 gradi e OVEST è a 180 gradi. Fra 0 e 180 dovrai perciò scegliere l'angolo che ti sembra più idoneo ad intercettare i nemici.

Hai armato il missile premendo "d"; ora il cursore lampeggia in basso a sinistra in attesa dell'angolo; scrivi perciò un numero fra 0 e 180 e premi "enter", il missile partirà con quella angolazione. Sullo schermo vedrai un punto nero lampeggiare che indica il volo del missile. Se in qualsiasi momento, dopo il decollo, vuoi cambiare direzione al missile, premi il tasto "c", il cursore lampeggiante attende il numero del missile che deve cambiare rotta: scrivi perciò un numero da 1 a 20 e batti "enter". Il cursore lampeggiante attende ora un angolo fra 0 gradi e 180 gradi in cui variare la rotta di volo: scrivi perciò un numero fra 0 e 180 e batti "enter"; vedrai il puntino nero, che rappresenta il missile, cambiare rotta, secondo le nuove istruzioni.

2) POSTAZIONE LASER

Come ben saprai, l'uso del Raggio Laser comporta un grande dispendio di energia. L'energia a tua disposizione è indicata dal numero in basso a sinistra sullo schermo.

Questa energia va aumentando con il passare del tempo.

Ogni colpo sparato con il Laser ti comporterà, naturalmente, una diminuzione di tale indice.

Per usare il Laser premi il tasto "L" per attivarlo; il cursore lampeggiante aspetta ora l'angolo di tiro (calcolato come per i missili): scrivi perciò un numero da 0 a 180 e batti "enter". Il cursore lampeggiante aspetta ora la potenza di fuoco, che varia da 1 a 20. Scrivi perciò un numero da 1 a 20 e batti "enter".

Attenzione alla potenza di fuoco! Più è alta e più è lungo il tiro, ma anche maggiore sarà la potenza assorbita.

Con meno di 120 MWatt di energia il Laser non spara.

Il gioco ha termine quando tutti i nemici saranno distrutti, o quando un nemico tocca terra. Ma contiamo sulla tua abilità perché questa seconda ipotesi non si verifichi. Good Luck.

SUPERDIZIONARIO

Non tutto, ma di tutto, questo può essere il filo conduttore per presentarvi questo programma.

Materialmente ci era impossibile fornirvi un dizionario completo Italiano-Inglese Inglese-Italiano; abbiamo, comunque, fatto del nostro meglio per spaziare nel maggior numero possibile di campi dello scibile umano...e scusateci se non ci trovate mamma o computer!!!!

Il menù iniziale del programma presenta 2 opzioni:
1) Inglese-Italiano
2) Italiano-Inglese

Scegliete battendo il tasto 1 o 2 l'opzione che volete utilizzare; a questo punto, in fondo allo schermo, il cursore vi chiederà di scrivere la parola da tradurre; scrivetela e battete "enter". Se la parola è memorizzata comparirà sul video la traduzione. Un'opzione "Copia su stampante s/n" vi permetterà, battendo "s" di fare una copia scritta, battendo "n" di passare oltre.

L'opzione successiva sarà "Altro termine s/n". Battete "s" per continuare la consultazione, battete "n" per uscire dal programma.

SEQUENCER

Hai buona memoria?

Con questo gioco puoi facilmente scoprirlo.

Il computer ti mostrerà delle sequenze di rettangoli colorati, che tu, al tuo turno, ed entro un tempo massimo, dovrai ripetere.

I comandi sono semplici:
il tasto 1 comanda il rettangolo Rosso
il tasto 2 comanda il rettangolo Blu
il tasto 3 comando il rettangolo Verde
il tasto 4 comanda il rettangolo Giallo

Il gioco si divede in turni: al primo turno il computer ti mostrerà un solo colore che tu dovrai ripetere.

Al secondo turno ti mostrerà il colore del primo turno più un altro colore, che tu dovrai ripetere...e così via fino a che la memoria ti aiuta.

P.S. Ai più smemorati consigliamo di mangiare pesce.

LINES

Astuzia, abilità, tempismo, sono le caratteristiche richieste per vincere in questo gioco.

Chiudi con la tua linea l'avversario, non dargli spazio vitale, questo è ciò che devi fare per sopravvivere.

Lo scopo del gioco è, infatti, di fare urtare il tuo avversario contro il bordo che delimita il campo; ma attento..!!

Non tornare sui tuoi passi.

Puoi giocare da solo contro il computer, o con un amico:
Premendo il tasto 1: selezioni il giocatore 1, umano o computer

Premendo il tasto 2: selezioni il giocatore 2, umano o computer
Premendo i tasti 3/4: selezioni la velocità del gioco
(il centro è più veloce)
Premendo il tasto 0: inizi a giocare.

Appena premuto il tasto 0 compare un rettangolo bianco (campo di gioco) in cui due linee vanno allungandosi sempre più. Puoi muovere la tua linea facendola curvare, a destra o a sinistra, o facendola andare in alto o in basso.

MOVIMENTI:

Giocatore 1	Direzione	Giocatore 2
Tasto "d"	ALTO	Tasto "j"
Tasto "x"	BASSO	Tasto "m"
Tasto "caps shift"	SINISTRA	Tasto "symbol shift"
Tasto "z"	DESTRA	Tasto "space"

SUGGERIMENTI:
se rimani chiuso in uno spazio delimitato, non arrenderti, vendi cara la pelle e comincia a girare il più possibile a ridosso delle linee che ti tengono prigioniero; può essere una lenta agonia, ma il tuo avversario potrebbe, nel frattempo, compiere qualche errore.

Meditate gente meditate!!!!!

BIORITMI

Secondo alcuni studiosi, la vita dell'uomo non è influenzata dagli Astri, come comunemente si credeva fino dall'antichità, ma da tre cicli vitali, che hanno inizio al momento della nascita.

Non staremo qui a farvi un trattato sulle cellule dell'ectoderma, del mesoderma, dell'entoderma, che sono le cellule che regolano i tre cicli.

Vi diremo solo che i tre cicli hanno una durata, in giorni, diversa fra di loro:
Ciclo fisico 23 giorni
ciclo emotivo 28 giorni
ciclo intellettivo 33 giorni.

Come vedrete poi, utilizzando il programma, ogni ciclo ha due fasi: di alta e bassa potenzialità.

Si usa chiamare Bioritmo Positivo la fase di alta potenzialità.

Si usa chiamare Bioritmo Negativo la fase di bassa potenzialità.

Ogni ritmo, positivo o negativo, dura la metà di un ciclo.

Questa curva sinusoidale, che potrete vedere con l'opzione "grafici" del programma, presenta dei giorni "critici":
questi giorni sono il Primo giorno di ogni ciclo, e il Primo giorno del ritmo negativo.

Per cui i giorni critici sono:
Ciclo Fisico: Primo, Dodicesimo o Tredicesimo
Ciclo Emotivo: Primo e Quindicesimo
Ciclo Intellettivo: Primo e Diciassettesimo o Diciottesimo.

Passiamo ora al programma.

Inserisce, come da richiesta del computer, la tua data di nascita:

1) scrivi il numero del giorno e batti "enter"
2) scrivi il numero del mese e batti "enter"
3) scrivi il numero dell'anno e batti "enter"
(l'anno va scritto integralmente: es. 1953)

Inserisci, ora, il mese e l'anno di cui vuoi il Bioritmo:
1) scrivi il numero del mese e batti "enter"
2) scrivi il numero dell'anno (es. 1985) e batti "enter".

Comparirà a questo punto una schermata divisa in colonne:
1) colonna G = giorni del mese
2) colonna Fis = Ciclo Fisico
3) colonna Emo = Ciclo Emotivo
4) colonna Int = Ciclo Intellettivo
5) colonna Media = media ponderata del vostro stato generale tenendo conto dei tre indici precedenti.

In fondo alla schermata, comparirà la scritta Scroll?: battendo "n" si esce dal programma, battendo "s" o "y" lo scroll verticale del video visualizzerà i rimanenti giorni del mese, che precedentemente non comparivano sullo schermo e l'opzione per i Grafici.

Se batti "n" ti chiederà se vuoi un altro mese, se batti "s" ti chiederà quale grafico vuoi.
Batti "f" per il grafico del ciclo Fisico
Batti "e" per il grafico del ciclo Emotivo
Batti "i" per il grafico del ciclo Intellettivo

Con il grafico prescelto comparirà l'opzione:
Copia su Stampante s/n
Se batti "s" copi lo schermo su Stampante
Se batti "n" ti comparirà l'opzione: Altro Bioritmo s/n
Se batti "s" ti chiede quale grafico vuoi (vedi sopra)
Se batti "n" ti chiede: ALTRO MESE s/n
Se batti "s" ti chiede data di nascita (vedi sopra)
Se batti "b" si blocca il programma.

Ora, sapendo le date di nascita dei calciatori, puoi farti anche la Bioschedina.

(Lato B: TI 99/4A. Le spiegazioni dei programmi appariranno direttamente in video).

Per caricare i programmi inserite la cassetta nel registratore e scrivete OLD CS1 seguito dal tasto ENTER. Apparirà la scritta REWIND CASSETTE TAPE, THEN PRESS ENTER, premete quindi il tasto ENTER, apparirà ora il messaggio PRESS CASSETTE PLAY THEN PRESS ENTER che significa: premi il tasto PLAY del registratore e quindi ENTER: avviate dunque il registratore in play e premete di nuovo ENTER.

Quando il computer avrà terminato di caricare il primo programma fermate il registratore e scrivete RUN seguito come sempre da ENTER per lanciarne l'esecuzione.

Quando avrete terminato di usare un programma resettate il computer spegnendolo per un momento, dopodiché ripetete le operazioni di caricamento sopradescritte.

Numeri già apparsi:

VIDEOTECA COMPUTER N. 1
per i possessori di Commodore 64

Nel manuale n. 1: I tasti funzionali del Commodore 64 - Pseudocodice e programmazione Basic - Il joystick.

Nella cassetta n. 1: Slalom - Slot Machine - Bilancio familiare - Briscola - Domino.

VIDEOTECA COMPUTER N. 2
per i possessori di Commodore 64

Nel manuale n. 2: Il basic più veloce - Una migliore gestione del video per il 64 - Disegnare con tastiera e joystick - Pseudocodice: 2ª lezione.

Nella cassetta n. 2: Tennis 3d - Totocalcio - Gestione magazzino - Wargame - Colour search.

VIDEOTECA COMPUTER N. 3
per i possessori di Commodore 64

Nel manuale n. 3: I cicli annidati del Basic - Come sviluppare un programma (Squash e Trampolino) - Conosci il tuo CBM 64?

Nella cassetta n. 3: Starway - Easyword - Poker - Forza 4 - Test Quoziente Intellettuale.

VIDEOTECA COMPUTER N. 4
per i possessori di Commodore 64

Nel manuale n. 4: Pseudocodice: Istruzioni read e data - Figura richiama - Grafico a barre. Il basic del CBM 64. I linguaggi di programmazione. Come personalizzare i programmi.

Nella cassetta n. 4: Dieta - Calorie dei cibi - Visischool - Sink - Hat in the ring.

PLAY ON TAPE N. 1
per i possessori di VIC 20

Nel manuale n. 1: I tasti funzionali del VIC 20 - Pseudocodice e programmazione Basic - Il joystick.

Nella cassetta n. 1: Totocalcio - Air attack - Cervellone - Inferno 3D - Bilancio familiare.

PLAY ON TAPE N. 2

Nel manuale n. 2: Il basic più veloce - I caratteri speciali del VIC 20 - Disegnare con la tastiera e il joystick - Pseudocodice: 2ª lezione.

Nella cassetta n. 2: Test per misurare il Quoziente intellettuale - Easyword - Caccia al tesoro - Gestione Magazzino - Formula 1.

PLAY ON TAPE N. 3
per i possessori di VIC 20

Nel manuale n. 3: I cicli annidati del Basic - Come sviluppare un programma (Corsa automobilistica - Piranha) - L'interfaccia sconosciuta - Conosci il tuo VIC 20?

Nella cassetta n. 3: Sette e mezzo - Dieta - Calorie dei cibi - Gangster - Costi chilometrici.

PLAY ON TAPE N. 4
Per i possessori di VIC 20

Nel manuale n. 4: Pseudocodice: Istruzioni read e data - Figura richiama - Grafici a barra. Il basic del VIC 20 - I linguaggi di programmazione - Come personalizzare i programmi.

Nella cassetta n. 4: G.O MO-KU - Calcolatrice - Golf - Vic Tab - Cabala e sogni.

COMPUTING VIDEOTECA N. 1
Per i possessori di Sinclair
SPECTRUM ZX

Nel manuale n. 1: Pseudocodice e programmazione basic: 1ª e 2ª lezione - La gestione dei canali dello Spectrum - Come farsi una cassetta di "Subroutines".

Nella cassetta: Wargame - Gestione del magazzino - U.F.O. - Helibomber.

COMPUTING VIDEOTECA N. 2
per i possessori di Sinclair
ZX SPECTRUM

Nel manuale n. 2: I cicli annidati del Basic - Come sviluppare un programma (Gara di sci) - Variabili interne e gestione del video nello Spectrum.

Nella cassetta n. 2: Frog Race - Pac Chian - Torre Laser - Dieta - Calorie dei cibi.

COMPUTING VIDEOTECA N. 3
per i possesori di ZX SPECTRUM e
TI-99/44A

Nel manuale n. 3: Pseudocodice 4ª lezione - Il Basic dello Spectrum - I linguaggi di programmazione - TI Super Sound.

Nella cassetta n. 3. Lato ZX Spectrum: Buio (48 k) - Anatomia (48 K) - Brain (48 K) - Visischool (16 K) - Escape (16 K)
Lato TI-99/4A: Poker - Planets explorer - Char designer - Forza 4 - Plat (extended Basic).

COMPUTING
VIDEOTECA
4
ZX SPECTRUM
• MIXXIL
• DIZIONARIO DI INGLESE
• SEQUENCER
• LINES
• BIORITMO
TI 99/4A
• SPACE WAR
• SLOT MACHINE
• AGENTE SEGRETO
• DIETA
• CALORIE
EV EDITORIALE VIDEO